REMARKS

This application has been carefully reviewed in light of the Office Action dated May 27, 2005. Claims 1, 2, 4 to 16 and 25 to 43 are pending in the application, of which Claims 1, 7, 12, 25, 27 to 30, 32 to 34 and 41 to 43 are independent. Reconsideration and further examination are respectfully requested.

As an initial matter, Applicants thank the Examiner for the indication that Claim 16 contains allowable subject matter and would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Claims 1, 4, 5, 9, 11, 14, 17 and 21 to 23 were rejected under 35 U.S.C. § 103(a) over U.S. Patent No. 5,668,996 (Randisky) in view of U.S. Patent No. 6,515,756 (Mastie). Reconsideration and withdrawal of these rejections are respectfully requested.

Interfacing peripherals to video game consoles is problematic. A main reason is because software for video game consoles is inherently unchangeable. Accordingly, the software, or modules thereof, cannot be updated to provide forward compatibility for new peripherals.

Also, in the case of other types of data processing apparatuses, large capacity hard disks may be provided for storing a multitude of software modules, which may include multiple drivers for different peripherals. When a new peripheral is provided, a new driver is typically also added to the hard disk. In contrast thereto, game consoles have limited Random Access Memory (RAM), and rely mainly on Read-Only Memory (ROM) devices for providing software to the consoles. Such ROM cannot be updated with new peripheral driver.

To overcome these limitations of prior systems, the present invention concerns generating a device driver in an information processing apparatus for a device connected thereto.

To do so, a device characterization file containing a characterization of the device is read from a memory. The device driver is then generated by using the device characterization file to configure a device model independent driver stored with an application on an unchangeable memory, such as a CD-ROM.

Turning to specific claim language, amended independent Claim 1 is directed to a computer implemented method for generating a device driver for an output device by an information processing apparatus, wherein the output device is connected to the information processing apparatus. The method includes the steps of: loading an application from a read-only memory, the application including a device model independent device driver; determining a model of the output device to which the application is intending to issue output commands; determining whether a device model dependent configuration data in a memory device matches the model of the output device; upon determining that the device model dependent configuration data in the memory device matches the model of the output device, reading the device model dependent configuration data from the memory device; and generating the device driver for the output device by configuring the device model independent device driver with the device model dependent configuration data.

In contrast, Randisky is seen to describe using a proxy device driver instead of a default CD audio driver provided by an operating system. The proxy device driver interfaces with an application program and a compound MCI device driver, such that the proxy device driver causes sound that would otherwise be rendered from data retrieved from a CDROM to be retrieved from sound files stored on a hard disk drive and rendered on a local sound card. The primary function of the proxy device driver is to translate between an interface it exposes to the

application program and an exposed interface of the compound MCI device driver (column 6, lines 4 to 7).

Accordingly, the application program described in Randisky is operable to use the default CD audio driver (a simple MCI device driver). The proxy device driver which replaces the default CD audio driver acts as a translator of commands from the interface of the application program, hence allowing the application program to still operate even though the audio files are now stored on the hard disk drive. That is, Randisky discloses replacing an existing device driver with a proxy device driver in order to replace one type of device with another type of device transparently as viewed from the application program.

However, Randisky fails to disclose several features of the present invention as claimed in Claim 1. Specifically, Randisky fails to disclose an application including a device model independent device driver, reading device model dependent configuration data from a memory device and generating a device driver for an output device by configuring the device model independent device driver with the device model dependent configuration data. In Randisky, the application and the proxy device driver are independent. In the present invention, the application is read from a read-only memory with the device model independent device driver included in the application program. Furthermore, as Randisky is concerned with conventional computing systems with large amounts of available storage memory, the proxy device driver is provided as a complete device driver. Therefore, the only processing necessary to use Randisky's proxy device driver is to replace an existing device driver with the proxy device driver. In the present invention, a device driver is generated using the device model independent device driver and device model dependent configuration data read from a separate memory device. Finally, as Randisky's proxy device driver simply replaces an existing device, Randisky

fails to disclose determining a model of the output device to which the application is intending to issue output commands and determining whether a device model dependent configuration data in the memory device matches the model of the output device.

Nothing in Mastie is seen to cure the deficiencies of Randisky. Mastie discloses determining print attributes for an input data file. However, it would be a mischaracterization of the disclosures of Mastie to equate the print attribute values with the present invention's device model dependent configuration data. This is because Matise discloses that:

> configuration files including predefined print attributes may be installed at locations in the network 2, e.g., storage 10. In preferred embodiments, a configuration file may define print attribute values for specific printers, print job, printer controllers 8a, b, c, and printer daemon (PD) types regardless of the printer controller 8a, b, c in which the printer daemon is executing. When the printer manager 6 receives a print job, the printer manager 6 determines the type of printer daemon to use, e.g., PS2AFP, D2AFP, TIFF2AFO, etc., based on the type of input data file, e.g., PostScript, ditoff, TIFF, etc. After determining the type of printer daemon (PD), the printer manager 6 would then select a printer daemon (PD) available in one of the controllers 8a, b, c. (Matise, Column 5, Lines 45 to 58).

That is, Mastie provides for using multiple printer daemons for each printer controller to achieve input data independence. The printer daemons provide this service by transforming input data files in various formats into a format for the printers based upon the attribute values of the input data file. However, Mastie is entirely silent on how the printer daemons manage device drivers for printers 12a to 12d (of Fig. 1). Specifically, Mastie fails to disclose loading an application

including a device model independent device driver and configuring the device model independent device driver with a device model dependent configuration.

Furthermore, at page 7, paragraph 28 of the Office Action, Snyders is cited as disclosing "generating a device for said mode[l] of said device by configuring said device mode[l] independent device driver code with said mode[l] depend[ent] configuration" at col. 2, lines 18 to 21, col. 6, lines 10 to 14, col. 8, lines 30 to 35, col. 9, lines 15 to 25, col. 51 , lines 12-15 and lines 35 to 37. Applicants respectfully submit that this is an inaccurate interpretation of the actual disclosures of Snyders. Snyders actually discloses:

> an application configured for running on the operating system and generating a source job comprising an output instruction file; and at least one output device having an output device driver for receiving the output instruction file for producing output. The method includes the steps of: providing a print processor in the form of an intermediate executable for operating on the output instruction file; **retrieving printer details for an identified printer driver name** from a memory location in a memory of the computer; **changing the printer driver name in the memory to a different driver of a different printer; saving the printer details of the identified printer driver name in the form of new printer information in a system registry; retrieving printer document properties of the saved printer details from memory; changing the retrieved printer document properties to match new driver settings**; saving the new printer document properties in the system registry; and allocating and initializing print processor data structures usable to execute a print job on the new printer. (col. 2, lines 25 to 50, emphasis added)

As can be seen from the above passage, Snyders discloses retrieving printer details for an identified printer driver name, changing the printer driver name to a different driver of a different printer, saving the printer details of the identified printer driver name in the form of new printer information in a system registry; retrieving printer document properties of the saved printer details from memory and changing the retrieved printer document properties to match new driver settings. Accordingly, Snyders actually teaches away from Applicants' method of loading an application from a read-only memory, said application including a device model independent device driver, reading device model dependent data, and generating a device driver for an output device by configuring the device model independent device driver with the device model dependent configuration data. This is because Snyders teaches modifying a print job based on a new printer driver's requirements rather than generating a device driver. This is possible in Snyders as an enhanced metafile is used as intermediate data between an application and a printer driver. That is, in Snyders, it is the data to be printed that is modified to match the printer driver. Accordingly, no device driver is generated as in amended Claim 1 of the present application.

In light of the deficiencies of Randisky, Mastie and Snyders as discussed above, Applicants submit that amended independent Claim 1 is now in condition for allowance and respectfully request same.

Amended independent Claims 21 and 22 are directed to an apparatus and computer program, respectively, substantially in accordance with the method of Claim 1. Accordingly, Applicants submit that Claims 21 and 22 are also now in condition for allowance and respectfully request same.

Claim 24 as amended is directed to a method of providing forward compatibility of device driver code of an unchangeable application with a plurality of device models. The application is stored on a read-only memory and not linked to other executable code. The method starts by including device model independent device driver code in the application. A model of an output device to which the application is desired to issue commands is next determined. Model dependent configuration data for the model of the output device is then read. Finally, a device driver for the model of the device is generated by configuring the device model independent device driver code with the model dependent configuration data.

Applicants submit that the discussion from above in regard to Claim 1 applies as well to Claim 24. Accordingly, Applicants submit Claim 24 is also in condition for allowance and respectfully request same.

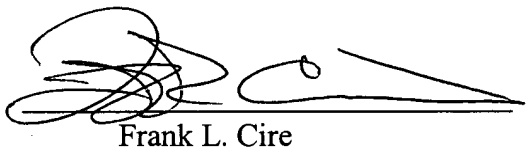
Independent Claim 28 is an apparatus claim corresponding to method Claim 24, whereas independent Claim 33 is a computer program claim corresponding to method Claim 24. Therefore, Applicants submit that Claims 28 and 33 are also now in condition for allowance and respectfully request same.

The other claims in this application are each dependent from one of the independent claims discussed above and are therefore believed allowable for at least the same reasons. Since each dependent claim is also deemed to define an additional aspect of the invention, however, the individual reconsideration of the allowability of each on its own merits is respectfully requested.

In view of the foregoing amendments and remarks, the entire application is believed to be in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

Applicants' undersigned attorney may be reached in our Costa Mesa, CA office at (714) 540-8700. All correspondence should continue to be directed to our below-listed address.

Respectfully submitted,



Frank L. Cire
Attorney for Applicants
Registration No. 42,419

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3800
Facsimile: (212) 218-2200

CA_MAIN 100471v1